Evaluating the Reliability of Explainable Machine Learning for Spacecraft Actuator Fault Detection

Pranav Narayan*, John R. Martin †
University of Maryland, College Park, MD, 20742

LIME and SHAP are widely used algorithms for fault detection classifier explanations, but their specific reliability, validated against known ground truth, in safety-critical aerospace applications has not been rigorously assessed. This study builds and employs a Basilisk anomaly simulation infrastructure to evaluate both methods on spacecraft actuator faults, varying fault timing, magnitude, and subsystem. Random Forest classifiers are used for detection of thruster and reaction wheel degradation from coupled physics-based input features. Analysis shows both LIME and SHAP achieve accurate feature attribution with severe faults, and performance degrades with fault subtlety. In addition, both methods are highly sensitive to classifier input and performance, yielding high performance in well-trained scenarios but worse performance under poor classification ability. SHAP is also seen to more regularly prioritize component-specific features than LIME, which may become desirable as correlation between features becomes stronger. Overall, these findings establish a baseline for applying interpretations of fault detection models to aerospace systems that can enhance downstream operator decision making.

I. Introduction

As spacecraft autonomy improves and machine learning-based algorithms play increasingly larger roles in spaceflight, it becomes more important to assess their reliability. In particular, algorithms used by operators to make crucial decisions must be transparent and certifiable without extensive excess analysis by a human. Explainable Artificial Intelligence (XAI) constitutes a class of algorithms that attempt to address and alleviate the "black-box" nature of many machine learning algorithms. Two approaches to addressing this problem are *explaining* and *interpreting* a model's output [1]. *Interpretability*, in this context, is typically defined as an ability to understand the underlying functionality of a model; in contrast, *explainability* is the ability to convey the reasoning of a model's output despite the incomprehensible mechanisms that produced it. Linear classifiers, for example, are *interpretable*, because the components of their learned weight vectors are directly indicative of reasoning, while a neural network's weight matrices are uninterpretable and require an *explanation* algorithm to understand.

XAI algorithms have been applied to a wide range of fields in the literature; Kalasampath et. al [2] alone surveys applications across environmental science, social media, education, finance, law, cybersecurity, agriculture, and healthcare. Algorithms exist that are both model-agnostic and model-specific—e.g. surrogate model interpretation algorithms or gradient-based methods applied to neural networks, respectively [3]. Applications to aerospace exist as well in the literature, though relatively few: Degas et. al [4] present a survey of applications to Air Traffic Management, for example, and Keneni et. al [5] propose an XAI algorithm for autonomous UAV path planning decision transparency. An extension to anomaly detection in spacecraft telemetry is proposed in Cuéllar et. al [6], who perform a thorough analysis of anomaly detection algorithms to set up a cursory application of XAI to one such algorithm.

This work expands the application to the anomaly detection problem for spacecraft telemetry, employing LIME and SHAP, two popular XAI *explanation* algorithms. A thorough analysis is performed to provide a rigorous quantitative benchmark for both algorithms' behavior, evaluating performance against a broad range of scenarios and fault characteristics in a spaceflight context and creating a baseline for reliability across the scope of the spacecraft anomaly detection problem.

II. Methods

While both LIME and SHAP fall under the class of Additive Feature Attribution explanation algorithms, which provide estimations in the form of weight vectors with entries corresponding to features, the methodologies they use to

*Undergraduate Student, Department of Aerospace Engineering, 3179 Glenn L. Martin Hall Bldg. 088, 4298 Campus Drive, AIAA University Student Member 1838698, pnaraya1@terpmail.umd.edu.
† Assistant Professor, Department of Aerospace Engineering, 3184 Glenn L. Martin Hall Bldg. 088, 4298 Campus Drive, AIAA Member

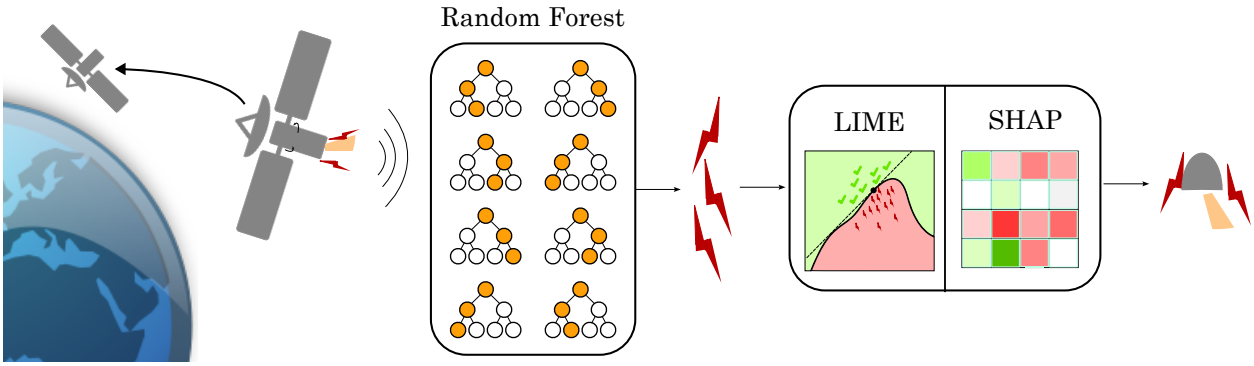

Fig. 1 Depiction of explanations in a rendezvous scenario under faulted thrusters.

calculate those weights differ significantly.

A. LIME

LIME—Local Interpretable Model-agnostic Explanations—is an approach to explaining individual predictions of any given complex machine learning model by interpreting a surrogate linear model that approximates the more complex model [7]. Because it relies on the interpretability of this unrelated surrogate model, LIME falls under the class of model-agnostic XAI algorithms.

LIME requires two inputs: a model f and an instance x, creating perturbed samples z_i by adding noise to a fixed number of random subsets of nonzero features in x. Each perturbed sample z_i is passed into the model f, yielding labels for each sample, as well as a value $\pi_x(z_i)$ that quantifies its distance from x, typically defined as:

$$\pi_x(z) = exp(-\frac{||x - z||_2}{\sigma^2}) \tag{1}$$

where σ is a hyperparameter, commonly set to $\sigma = 0.75\sqrt{d}$, that determines the relative influence of closer samples to further samples. These quantities are then used to train the surrogate linear model g in two steps: running LASSO regression on the samples z_i and their predictions $f(z_i)$, then applying least squares regression to the K features with the largest weights in the results of the LASSO regression.

In the original regression, the following loss function is minimized:

$$\mathcal{L}(f, g, \pi_x) = \sum_{\forall z}(\pi_x(z)(f(z) - g(z))^2) + \lambda||w||_1 \tag{2}$$

where w is the resultant weight vector for g. This weights the squared error by distance, allowing the LASSO regression model to prefer weight vectors that more correctly regress samples closer to x. The L1 regularization term additionally encourages sparsity in the final weight vector. The vector w that is found from the least square regression is then associated with a linear approximation g of the behavior of the model f around the instance x. Each component describes the impact of its associated feature on g. w, then, is the explanation for the prediction made by f on x.

Practically, while the algorithm is relatively fast and provides strong interpretability, LIME's approximations erode in high-dimensional, highly nonlinear, or tightly constrained applications due to the limitations of the linear surrogate and random perturbations.

B. SHAP

SHAP—SHapley Additive exPlanations—is another approach to XAI that provides similar weight vector explanations [8]. Compared to LIME, SHAP's strengths are its theoretical founding, consistency guarantees, and global attributions, affording it an additional measure of trust at the cost of computational feasibility. The bases of SHAP are the Shapley values of cooperative game theory, which define a feature's importance as the impact on the model's prediction when that feature is removed from consideration. Mathematically, the explanation algorithm for SHAP is as follows:

$$\phi_i(f,x) = \sum_{z' \subseteq x'} \frac{|z'|!(M-|z'|-1)!}{M!}[f_x(z') - f_x(z'\backslash i)] \tag{3}$$

where M is the number of simplified input features and $|z'|$ is the number of nonzero features in z'. This is essentially a combinatorics problem: it sums the outcome of a model across all possible feature subsets that contain a given feature, and subtracts the outcome of each of those subsets without that given feature. In this case, it adds the assumption that absent features are set to the expected value of that feature in a global background dataset. This methodology gives SHAP a global context despite being a local explanation algorithm. The output weights are then the contribution of a given feature to a prediction.

Finding an exact solution to this equation is expensive, with a computational complexity of $O(2^n)$. However, algorithms exist to make SHAP computationally feasible. Particularly relevant to this study is TreeShap, which leverages the structure of tree-based models to simplify the time complexity required to explain them to polynomial time [9]. At a node in the tree, if it has not yet encountered the feature that node is splitting on, the tree recursively finds the contribution of training samples that followed each path, and uses that to generate an expected value of the prediction. After a feature is encountered, the value of that feature in the instance being predicted can be fixed for all future nodes splitting on that feature. This allows features' contributions to only be calculated once, and reduces the number of branches the algorithm needs to travel. Each contribution is scaled by the Shapley kernel weight and the probability of a path being taken, and then summed to get the SHAP values.

While SHAP provides additional guarantees relative to LIME, this does come at the cost of complexity; like LIME, though, SHAP suffers from an inability to handle feature correlation and tightly constrained applications. SHAP typically also has sensitivity to a well-chosen background dataset.

III. Experiments

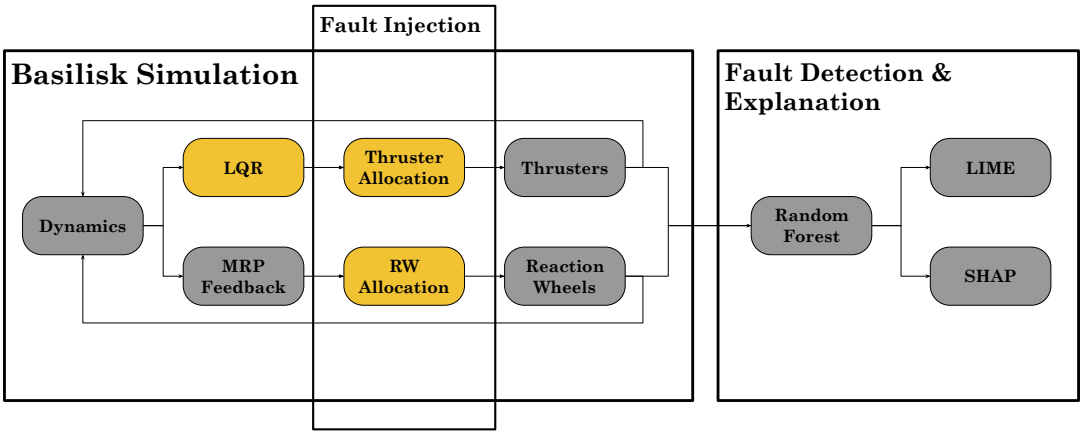

Fig. 2 Simple flowchart of the simulation and machine learning modules used in each experimental instance. Grey modules are either built-in or taken from popular Python libraries; gold modules are custom for this work.

This work builds a fault detection pipeline intended to closely mimic a real spacecraft for the purposes of testing LIME and SHAP. This includes a series of high-fidelity simulations in the Basilisk astrodynamics simulation framework [10], with specialized modules for fault injection, and the resultant telemetry is then passed through a Random Forest detection model to which LIME and SHAP can be applied. Fig. 2 presents the full telemetry pipeline for each experimental run, including the modules unique to this study.

A. Simulation Framework

Experiments are carried out on a spacecraft in two different dynamical scenarios of differing complexities. The spacecraft is identical in both, with 6 total thrusters pointing along each positive and negative body frame axis and 3 total reaction wheels with spin axes about each body frame axis. The reaction wheels are torque-limited to 2.0 $N \cdot m$,

selected to exceed controller demands. The thrusters are limited to 25.0 N thrust and a minimum on-time equal to the simulation timestep of 0.1 s. the initial attitude of the spacecraft, σ_{BN}, is randomized, while the initial angular velocity, $\Omega_{BN,B}$, is set to zero.

1. Slewing Scenario

The simpler dynamical scenario places the spacecraft in a circular, equatorial orbit with a radius of 7000 km, running for 0.25 orbits while combating a constant external torque of 0.001 $N \cdot m$ across all three axes. This torque requires consistent demands from the reaction wheel to maintain a desired attitude, enhancing the visibility of faults. From the random initial attitude, Basilisk's built-in *mrpFeedback* controller is tasked with achieving and maintaining an attitude that aligns the body frame with the inertial frame: $\sigma_{BN} = [0.0 \quad 0.0 \quad 0.0]^T$. The telemetry generated by this simulation consists of commanded and applied torque vectors for each timestep.

2. Rendezvous Scenario

The more complex dynamical scenario features two spacecraft: a target and a chaser. The chaser uses its thrusters to attempt a rendezvous approach with the inactive target, beginning at a randomized point on a 100m-offset sphere around the target, while its reaction wheels attempt to point towards the target. The final desired state is a 1m radial offset with the chaser body frame x-axis pointing at the target. A Linear Quadratic Regulator (LQR) is employed to achieve the translational rendezvous. As the target spacecraft is kept in a circular orbit, with radius 6878 km, the dynamics are assumed to follow the Clohessy-Wiltshire formulation for relative motion. To simplify controller design, both the state error and control effort penalty matrices were assumed to be diagonal; the latter was defined as a scaled identity matrix, the former as block-diagonal with velocity error weighted three orders of magnitude more than position error, and x-velocity weighted double y- and z-velocity to account for controllable secular drift. A selection of candidate trajectories in simulation are visible in Fig. 3.

The rendezvous scenario is further split into two cases. The first case maintains uniform initial conditions—i.e. initial chaser offset vector and attitude—across all training trajectories and the test trajectory, while the second case randomizes initial conditions across the training and test trajectories such that each is unique. The random-IC case therefore presents an increase in complexity over the fixed-IC case.

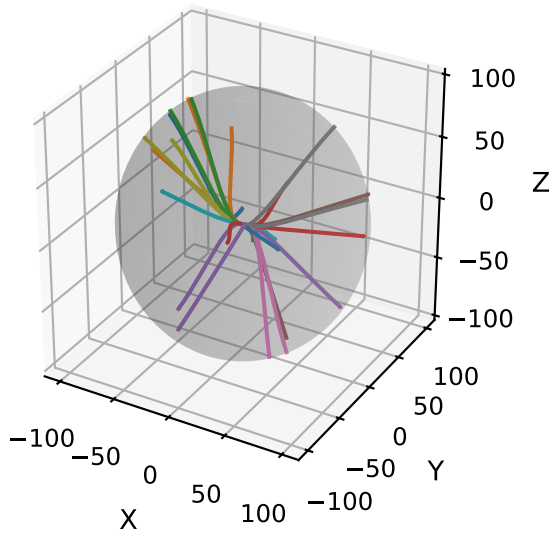

Fig. 3 25 randomized nominal trajectories in the rendezvous scenario, shown in the target spacecraft's Hill frame.

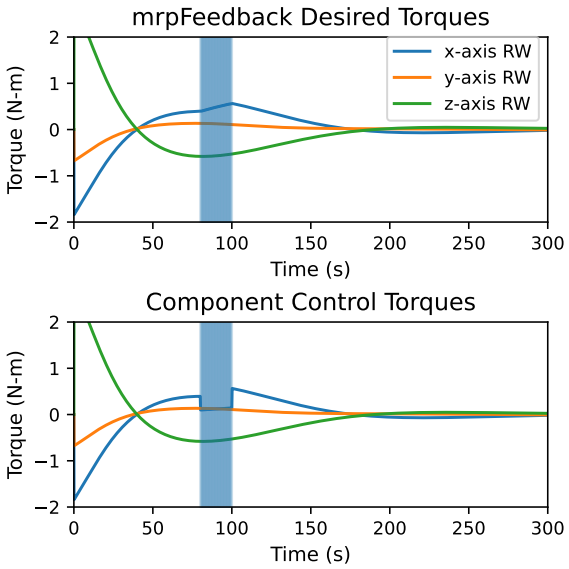

Fig. 4 Candidate rendezvous trajectory with x-axis RW fault.

The telemetry generated by this simulation consists of the chaser's position and velocity in the target's Hill frame,

normalized commanded body frame force and torque, and normalized applied body frame force and torque, for each timestep. Normalization is done relative to the maximum output for the corresponding actuator.

B. Fault Injections

While Basilisk contains built-in modules for actuator modeling and simulation, these modules do not readily provide interfaces for fault injection. This work has therefore built two new modules, placed between controller and actuator modules, to modify commands in such a way that the result appears as a fault. These modules both allocate commanded actuation to the necessary components and modify those commands, applying the same fault architecture to reaction wheels and thrusters. A fault, assuming the actuator type is known, can generally be represented as the following four-tuple:

$$f_i = \left(\delta_i,\ t_i^s,\ t_i^e,\ \mathbf{a}_i\right) \tag{4}$$

where δ_i is degradation fraction—i.e. fraction actuator effectiveness—t_i^s and t_i^e are the onset and conclusion of the fault, respectively, and $\mathbf{a}_i$ is the axis on which the faulted actuator is located. Its impact on the commanded actuation is then:

$$\mathbf{u}_{\text{applied},\,i}(t) = \max(\delta_i,\ d_i(t))(\mathbf{u}_{\text{cmd},\,i} \odot \max_{\odot}(\mathbf{0}_{3\times1} + \mathbf{d}_{i,\,3\times1}(t),\ \mathbf{1}_{3\times1} - \mathbf{a}_i) \tag{5a}$$

$$d_i(t) = \mathbb{1}(\neg(t_i^s \le t \le t_i^e)) \tag{5b}$$

where $max_{\odot}$ is an elementwise maximum. In other words, during the defined interval, the fault applies a linear degradation factor to the actuator command along a selected axis. An example trajectory with an x-axis reaction wheel fault with $\delta_i = 0.25$ is visible in Fig. 4.

Only one fault definition is applied to a given trajectory. It also allows for simple definition of a ground-truth fault for explanation evaluation: the features most relevant to the fault are defined as the commanded and actual actuations along axis $\mathbf{a}_i$. In the case of reaction wheels, this amounts to two features; for thrusters, because both positive and negative components exist per-axis, this amounts to three features. Restricting the fault space to this extent allows for isolated assessment of LIME and SHAP attributions independently against a broad range of parameters, and ensures the conflicting influences of more complex fault signatures do not reduce the significance of any conclusions.

C. Parameter Distribution

The results of this study include significant variance across simulation and model parameters, fault detection model performance, and feature engineering choices. Table 1 characterizes the variation introduced throughout simulation runs. The choice of faulted component type and faulted axis is also randomized. The full scope of this randomization is included only in test trajectories; t_i^s, $t_i^e - t_i^s$, and δ_i are uniformly spread across the given ranges during the training trajectories.

Value ranges for three parameters are demarcated to indicate increased difficulty for accurate attribution of a given trajectory: $\delta \ge 0.5$, $t_i^s \ge 250\,s$, and $|N| \ge 10$, where N is the set of included features in detection and explanation. Then, individual trajectories can be split into difficulty levels: *easy* instances exhibit one or zero of those parameters, *medium* instance exhibit two, and *hard* instances exhibit three. It is notable that the slewing scenario, due to its telemetry having fewer total features, will have zero *hard* instances.

D. Fault Detection and Explanation

To detect faults, this work employs a Random Forest model. This algorithm is selected to maintain a level of explanation simplicity in repeated experiments while still capturing the nonlinearities required to effectively detect faults in a high-dimensional, highly correlated feature space; it also enables the use of TreeSHAP. Each experimental run consists of 15 trajectories used as training data for the Random Forest, as well as one for testing. Each trajectory's fault definitions are unique, with the exception of identical values for $\mathbf{a}_i$ and actuator type per run. Only correct positive predictions are passed to the XAI algorithms, which are then performed per-instance. To aggregate metrics across all instances, the attribution of LIME and SHAP is treated as the mean attribution vector taken over all explained instances for a given trajectory. Plots and statistics, unless mentioned otherwise, are generated using 1000 faulted test trajectories of their scenario.

	Slewing	Rendezvous
δ_i	$[0, 1)$	$[0, 1)$
t_i^s	$[0, 500) \cap \mathbb{Z}$	$[0, 550) \cap \mathbb{Z}$
t_i^e	$[10, 510) \cap \mathbb{Z}$	$[10, 560) \cap \mathbb{Z}$
$t_i^e - t_i^s$	$[50, 500) \cap \mathbb{Z}$	$[10, 70) \cap \mathbb{Z}$
# excess features	$[0, 4] \cap \mathbb{Z}$	$[0, 19] \cap \mathbb{Z}$

Table 1 Test Trajectory Parameter Randomization Ranges

E. Explanation Quality

A set of predefined metrics for explanation quality must be chosen to effectively compare algorithm performance across many iterations. These metrics include a modification of Mean Reciprocal Rank and Cosine Similarity. Both compare the XAI attributions to the ground truth nominal attribution. This work modifies MRR by taking the maximum value, since only a select few features are actually desired and the use of the mean would suggest equal importance for irrelevant features. The modified definition is:

$$MRR(i) = \max_{d \in D} \left(\frac{1}{\text{rank}(d)} \right) \tag{6}$$

where D is the set of desired elements. It compares an ordering of features against a desired ordering, resulting in a discrete set of potential values, and outputs the maximum such value. Cosine Similarity is defined as the dot product of two vectors, in this case a desired and actual attribution vector, normalized by their magnitudes:

$$CS(\mathbf{A}, \mathbf{B}) = \mathbf{A} \cdot \mathbf{B}, \qquad \text{where } ||\mathbf{A}|| = ||\mathbf{B}|| = 1 \tag{7}$$

Cosine Similarity is continuous, unlike the modified MRR, and is therefore used primarily for analysis relative to the continuous dependent variables. Unlike MRR, though, it does not provide an explicit assessment of desired orderings.

The desired feature vectors are determined based on two desired evaluation criteria: residual-based attributions and component-specific attributions. The former accepts attributions to both the specific faulted component and its commanded actuation. Since the attribution vectors must have a magnitude of 1, the correct faulted component is weighted at $\frac{1}{\sqrt{1.25}}$ and the commanded actuation axis at $\frac{0.5}{\sqrt{1.25}}$. The latter accepts only the specific faulted component, weighted at 1. The weights are only used in analysis with Cosine Similarity. In a thruster fault, both components are faulted; however, only one can be labeled as the desired component, so this study chooses to calculate metrics for both and picks the component that yields the maximum value.

IV. Results

Explanation quality is generally seen to be high across scenarios, as visible by the high concentration at higher values in Fig. 5. LIME and SHAP rank at least one of the desired features top-3 73% and 82% of the time, respectively, in fixed-IC rendezvous. Increasing the complexity of the scenario causes a drop in MRR: LIME's top-3 percentage is 60% and SHAP's is 64% in random-IC rendezvous. The top-1 percentages show, however, that while LIME continues to follow that trend, SHAP does not. LIME's top-1 percentages are 45%, 36%, and 27%, in order of increasing complexity, while SHAP's are 41%, 49%, and 32%,

LIME and SHAP's expected behavior is that explanation quality degrades under subtle faults and high-dimensional feature spaces. The average Cosine Similarities in Table 2 show general degradation of explanation quality with increasing difficulty as defined. SHAP also sees a larger increase in performance in the *easy* instances. The trend suggests that fault severity and high-dimensional feature spaces are strong predictors of explainer performance.

The classifier is also dependent on difficulty, however, and LIME and SHAP's performance is not independent of the classifier. Across all three scenarios, only SHAP's Cosine Similarities actually parallel the performance of the Random Forest: despite slewing being a simpler scenario, both SHAP and the Random Forest perform better on the fixed-IC rendezvous scenario, unlike LIME. LIME's degradation then appears to be dependent on scenario complexity, while SHAP's degradation is strongly correlated with classifier degradation. This suggests that SHAP is more model-faithful while LIME is sensitive to local instabilities. Fig. 6 supports this conclusion, where it is visible that LIME is more

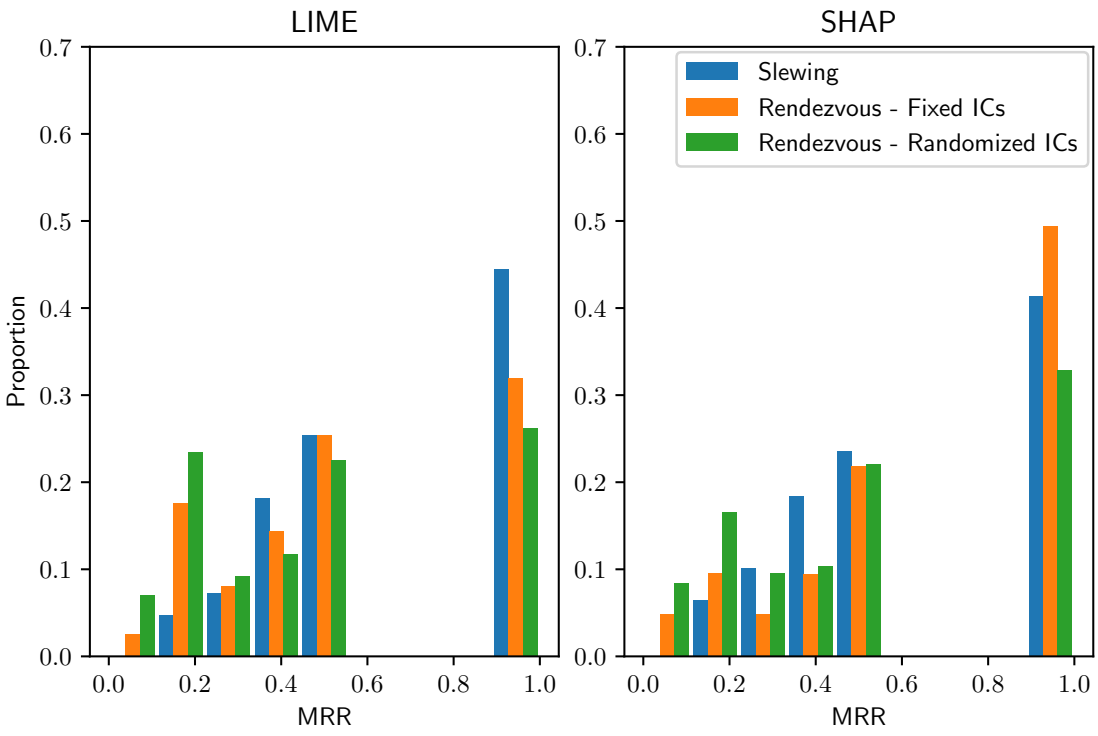

Fig. 5 MRR proportions for LIME and SHAP across all three scenarios.

		Easy	Medium	Hard
Slewing	LIME	0.79	0.84	N/A
	SHAP	0.77	0.74	N/A
	F1-Score	0.66	0.56	N/A
Rendezvous: Fixed ICs	LIME	0.73	0.62	0.52
	SHAP	0.81	0.73	0.72
	F1-Score	0.83	0.70	0.56
Rendezvous: Random ICs	LIME	0.63	0.50	0.48
	SHAP	0.70	0.53	0.50
	F1-Score	0.48	0.32	0.23

Table 2 Average Cosine Similarities for LIME and SHAP alongside the Random Forest detection model's F1-Score across all three scenarios and difficulties.

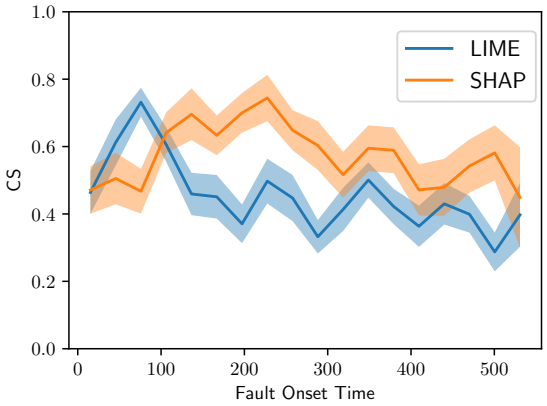

Fig. 6 Average Cosine Similarity vs Fault Onset Time for LIME and SHAP in the Fixed Rendezvous case, with 95% confidence interval.

strongly correlated with fault onset time. LIME's correlation coefficient with respect to fault onset time is -0.27, while SHAP's is only -0.03 and visibly less consistent.

Both LIME and SHAP show correlations to the Random Forest's F1-Score in the rendezvous cases, as visible in Fig. 7. The fixed-IC case shows less of this correlation—its high F1-Scores render the degradation less significant. Both only see rare occasions of high performance from LIME and SHAP in the absence of high Random Forest performance,

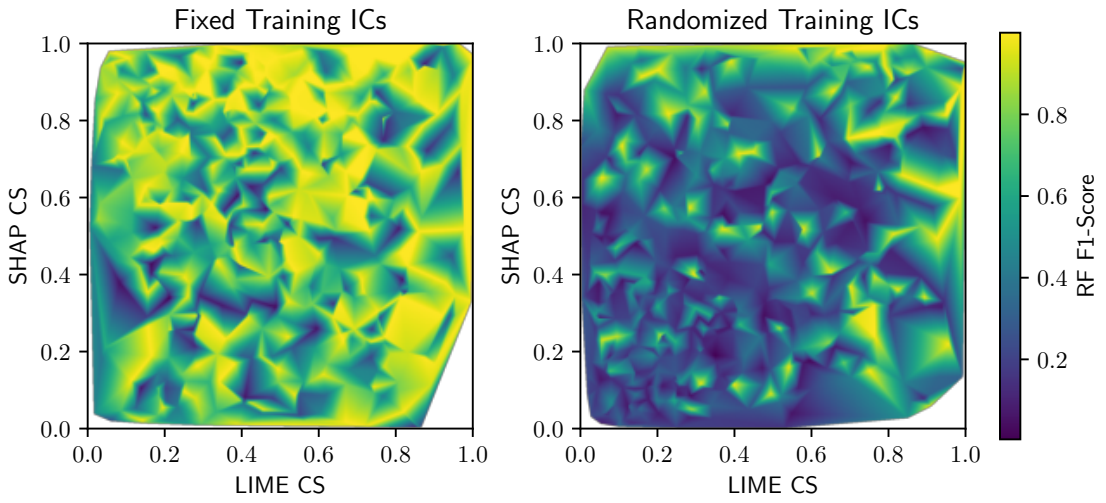

Fig. 7 Random Forest F1-Score vs LIME and SHAP Cosine Similarity—Rendezvous with Fixed and Randomized ICs.

indicating a strong dependence. Additionally, Fig. 7 indicates that there is a notable lack of instances—visible as empty spaces—resulting in both high-performing LIME and low-performing SHAP, particularly in the fixed-IC case. The converse is not as true, suggesting SHAP's increased robustness.

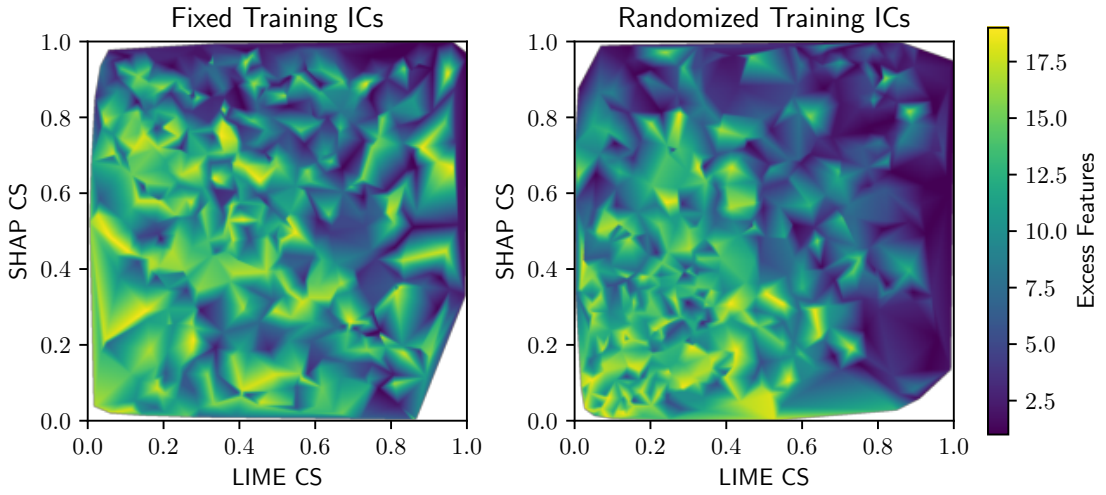

Fig. 8 Excess Features in Feature Space vs LIME and SHAP Cosine Similarity—Rendezvous with Fixed and Randomized ICs.

A final parameter is the number of excess features fed into the detection-explanation pipeline. This is partially encompassed into the difficulty levels; however, it displays a particularly strong correlation with classifier performance, as visible in Fig. 8. This correlation is visibly stronger than the F1-Score in both cases, and shows clearly that high-dimensional feature spaces and their inclusion of redundant information produce explanations of reduced quality.

Thus far, all Cosine Similarity values have used the residual-based attributions as the ground truth. It is worth exploring differences in overall performance if a component-specific approach is taken, as an operator is likely most interested in the failure of a specific component. Fig. 9 shows that SHAP maintains its consistent edge, and even improves it slightly in both rendezvous scenarios. SHAP's average Cosine Similarity for fixed-IC and random-IC rendezvous, respectively, are 0.1 and 0.04 higher than LIME's when calculated componentwise, and 0.08 and 0.03

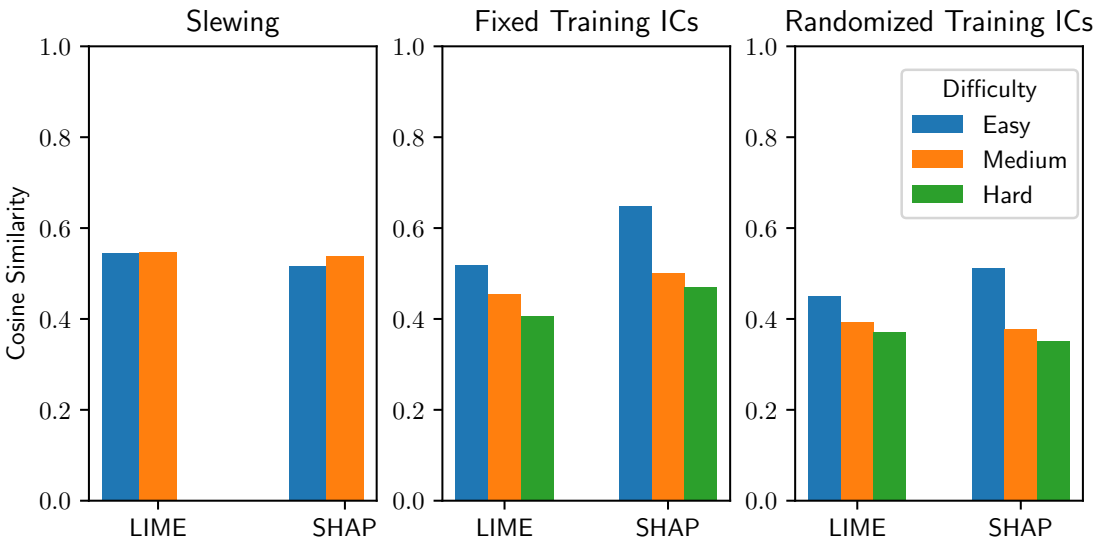

Fig. 9 Average Cosine Similarities, calculated componentwise, for LIME and SHAP across all three scenarios and difficulties.

higher with residual-based attributions. The averages are significantly lower than the residual-based attribution method: LIME and SHAP's residual-based averages are 0.67 and 0.71, respectively, while their componentwise averages are 0.48 and 0.54. This is to be expected due to the stricter definition of a "correct" explanation. SHAP's slight advantage in performance over LIME, though, appears to indicate its better ability to focus on the component.

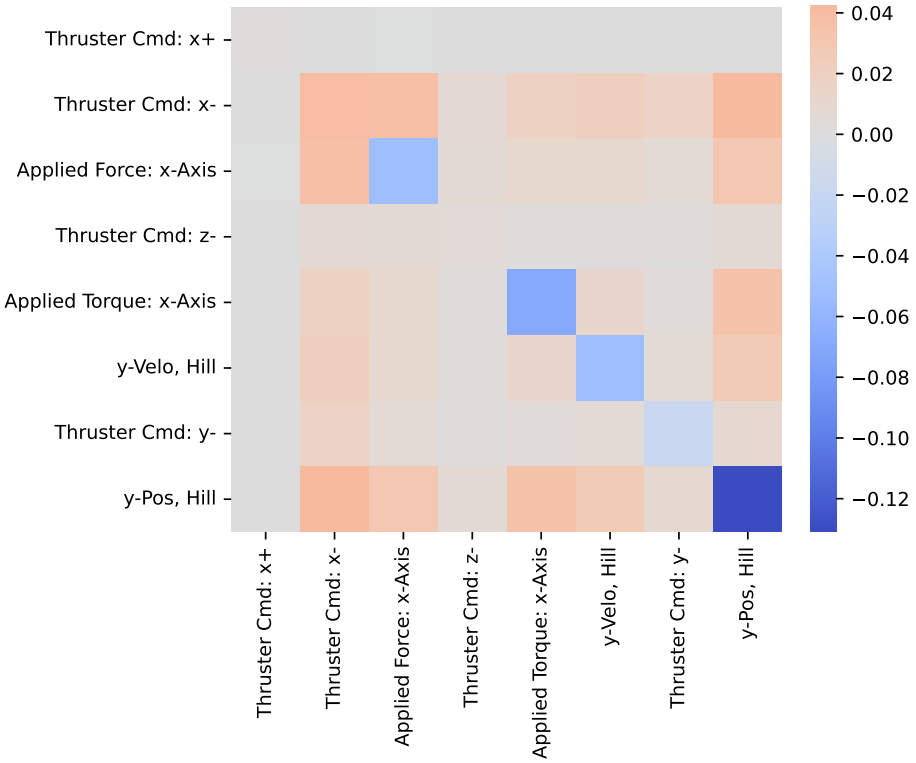

Fig. 10 SHAP interaction values for a candidate x-axis thruster fault in fixed-IC rendezvous.

Finally, SHAP has the ability to provide metrics for the relative interaction between different features and their impacts on the model output, known as *interaction values*. Fig. 10 is a matrix containing the contribution of each feature-feature interaction to the final output. The diagonals, displaying self-interaction and therefore the same substance as the final weight-vector, are expectedly large; the off-diagonal terms show the Random Forest's understanding of the physical relationships and dynamical coupling between features. Notably, the x-axis thruster command and applied force show slightly elevated interaction in opposite directions, which is exactly what the model is intended to notice in the form of their residual.

The model also captures an important feature of Clohessy-Wiltshire dynamics: a secular drift in Hill frame y-position that's most directly controllable by x-axis Hill frame thrust, as evident in the higher interaction value between those two features. This indicates that SHAP is capable of providing insight into the dynamical structure of a problem, and may even employ the underlying model's knowledge of dynamics in its explanations, exactly as desired. This heatmap does appear to indicate that the explanation provided by SHAP for this trajectory is suboptimal: the most significant feature is Hill y-position rather than commanded/applied force. However, the actual maximum value returned by SHAP is negative x-axis thrust for this instance. Rather than being suboptimal, then, this heatmap is a way to illustrate the importance of the combined influence of several feature interactions over one strong self-interaction, and highlights that SHAP is capable of capturing the complex interrelationships between features.

V. Conclusions and Future Work

This paper analyzes the performance of LIME and SHAP applied to anomaly detection in spacecraft telemetry. A simulation with customizable fault injections is built to enable parameter sweeps across fault signatures, and experiments are run under slewing and rendezvous scenarios experiencing thruster and reaction wheel degradation. Random Forest is employed as a fault detection algorithm that LIME and SHAP are tasked to explain for a wide variety of situations.

LIME and SHAP both produce high-quality explanations across all tested scenarios, maximized under simple dynamics and severe anomalies. However, both algorithms show noticeable drops in performance with fault subtlety, as well as heavy reliance on good classifier performance and feature space dimensionality. Evidence additionally suggests SHAP provides a more robust, practical explanation, though reliant on a good underlying model.

The results of this work demonstrate the viability of both LIME and SHAP applied to spacecraft anomaly detection. Additionally, it provides a baseline framework and analysis of both algorithms' performance against a known truth—crucially, grounded in high-fidelity simulation. However, this work limits itself to a relatively simple fault detection model and unenhanced configurations of LIME and SHAP. As the underlying model is seen to be one of the strongest predictors of explanation quality, future work should consider applying both algorithms, to the extent possible, to more sophisticated models—e.g. Neural Network- or Transformer-based architectures. Additionally, as LIME and SHAP's local explanation approach is inherently limited in its application to time-series data, many approaches exist to adapt both algorithms to improve performance. Such approaches are not employed in this work, and should be explored in future studies.

References

[1] Burkart, N., and Huber, M. F., "A Survey on the Explainability of Supervised Machine Learning," Vol. 70, 2021, pp. 245–317. https://doi.org/10.1613/jair.1.12228, URL https://www.jair.org/index.php/jair/article/view/12228.

[2] Kalasampath, K., Spoorthi, K. N., Sajeev, S., Kuppa, S. S., Ajay, K., and Maruthamuthu, A., "A Literature Review on Applications of Explainable Artificial Intelligence (XAI)," Vol. 13, 2025, pp. 41111–41140. https://doi.org/10.1109/ACCESS.2025.3546681, URL https://ieeexplore.ieee.org/document/10908240/.

[3] Devireddy, K., "A Comparative Study of Explainable AI Methods: Model-Agnostic vs. Model-Specific Approaches," , 2025. https://doi.org/10.48550/arXiv.2504.04276, URL http://arxiv.org/abs/2504.04276.

[4] Degas, A., Islam, M. R., Hurter, C., Barua, S., Rahman, H., Poudel, M., Ruscio, D., Ahmed, M. U., Begum, S., Rahman, M. A., Bonelli, S., Cartocci, G., Di Flumeri, G., Borghini, G., Babiloni, F., and Aricó, P., "A Survey on Artificial Intelligence (AI) and eXplainable AI in Air Traffic Management: Current Trends and Development with Future Research Trajectory," Vol. 12, No. 3, 2022, p. 1295. https://doi.org/10.3390/app12031295, URL https://www.mdpi.com/2076-3417/12/3/1295.

[5] Keneni, B. M., Kaur, D., Al Bataineh, A., Devabhaktuni, V. K., Javaid, A. Y., Zaientz, J. D., and Marinier, R. P., "Evolving Rule-Based Explainable Artificial Intelligence for Unmanned Aerial Vehicles," Vol. 7, 2019, pp. 17001–17016. https://doi.org/10.1109/ACCESS.2019.2893141, URL https://ieeexplore.ieee.org/document/8612916/.

[6] Cuéllar, S., Santos, M., Alonso, F., Fabregas, E., and Farias, G., "Explainable anomaly detection in spacecraft telemetry," Vol. 133, 2024, p. 108083. https://doi.org/10.1016/j.engappai.2024.108083, URL https://www.sciencedirect.com/science/article/pii/S0952197624002410.

[7] Ribeiro, M. T., Singh, S., and Guestrin, C., ""Why Should I Trust You?": Explaining the Predictions of Any Classifier," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Association for Computing Machinery, 2016-08-13, pp. 1135–1144. https://doi.org/10.1145/2939672.2939778, URL https://dl.acm.org/doi/10.1145/2939672.2939778.

[8] Lundberg, S. M., and Lee, S.-I., "A Unified Approach to Interpreting Model Predictions," *Advances in Neural Information Processing Systems*, Vol. 30, Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html.

[9] Lundberg, S. M., Erion, G., Chen, H., DeGrave, A., Prutkin, J. M., Nair, B., Katz, R., Himmelfarb, J., Bansal, N., and Lee, S.-I., "From Local Explanations to Global Understanding with Explainable AI for Trees," Vol. 2, No. 1, 2020, pp. 56–67. https://doi.org/10.1038/s42256-019-0138-9, URL https://pmc.ncbi.nlm.nih.gov/articles/PMC7326367/.

[10] "Welcome to Basilisk: an Astrodynamics Simulation Framework — Basilisk 2.9.0 documentation," , 2026. URL https://avslab.github.io/basilisk/.